RECOMMENDATION FRAMEWORK: IMPROVING AUTOMATED COLLABORATIVE FILTERING BY TRUSTED CATEGORY

By

Md Sadaf Noor
11121023

A thesis submitted to the Department of **Electrical and Electronic Engineering** in partial fulfillment of the requirements for the degree of
BSc in **Electrical and Electronic Engineering**

Electrical and Electronic Engineering
Brac University
December 2015

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. I/We have acknowledged all main sources of help.

Student's Full Name & Signature:

Md Sadaf Noor
11121023

Approval

The thesis/project titled “Recommendation Framework: Improving Automated Collaborative Filtering by trusted category” submitted by Md Sadaf Noor (11121023) of Fall, 2015 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of BSc in **Electrical and Electronic Engineering** on 20th December 2015.

Examining Committee:

Supervisor:
(Member)

____

Dr. Tarem Ahmed
Assistant Professor
Electrical and Electronics Engineering
BRAC University

Abstract/ Executive Summary

The key idea behind user user automated collaborative filtering is that it predicts item's rating based on similar users who share the same taste by rating item similarly. Automated collaborative filtering (acf) is proposed on a hypothesis that users with similar rating will also have similar rating in everything. People often agree on an idea, or on a group of ideas but it is very rare that agree on everything. So in this thesis we suggest that this is one of the main causes behind huge noises while working with a large number of neighbors in acf. In this thesis we are proposing a method where at the time of calculating acf we will only consider a subset of products based on their category that we trust based on users previous ratings. Usually when a developer has to build a recommendation system he has to start writing code from scratch like in the early days of web development, which consumes a lot of time and development energy so we tried to build a framework for our updated acf that provides the building blocks for a recommendation system as APIs. By describing models of their recommender in simple JSON format, within a few commands and using provided API support, programmers can easily get a customizable generic service running. All the input and output communications are done using RESTful APIs so that the system can communicate with any part of the whole system.

Keywords: Recommended System, Recommender, Trusted Category

Dedication

We would like to express the deepest and most sincere appreciation to our thesis supervisor Dr. Tarem Ahmed, Assistant Professor of EEE department, BRAC University for his guidance, stimulating suggestions and encouragement throughout the course of this work. Without his supervision and constant help this dissertation would not have been possible.

Table of Contents

List of Tables

Chapter 1

Introduction

1.1 Introduction:

Success of a product nowadays depends on how well we design our recommender and how well our recommendation system works as a whole. To achieve our goals in recommendation at business we have developed different techniques and algorithms. User user collaborative filtering is one of them. It tries to establish a relationship between users and their rating based on other similar users rating patterns. The main hypothesis of user user collaborative filtering is that the user who has a similar previous rating will also like a similar rating in future. User user collaborative filtering works pretty well but yet it fails to explain certain cases. For example what if user A likes only scifi and action movies and hates romantic movies, another user B likes action and romantic movies. User A and B have 80% common in action movies. So automated collaborative filtering algorithms will start suggesting romantic movies to user A. So in this thesis we are suggesting several ways to solve this problem by considering trusted category of movies.

1.2 Background of Recommendation

The idea of a recommendation system is very old. At the age of caveman, people started to find the recommended way to find by figuring out the foot marks on the land, or they started to distinguish the outcomes between poisonous and non poisonous food by seeing the outcomes of their peers, these can be the example of recommendation. These kinds of recommendations can be found even in other lower level animals. Ant genetically evolved to leave markers for other ants. So recommendation is a very old concept but in modern days recommendation systems play a vital role in the market. If we can recommend the right product to the right person, at the same time increase the sell of the product and it also saves valuable time for that customer. The sizes of these decision domains are frequently massive. Netflix has over 17,000 movies in its selection [5], and Amazon.com has over 410,000 titles in its Kindle store alone [4]. There has been a good deal of research over the last 20 years on how to automatically recommend things to people and a wide variety of methods have been proposed [6][7].[2] Not only improving algorithms is sufficient in improving recommendation systems. In his keynote address at the 2009 ACM Conference on Recommender Systems, Martin [8] argued that the algorithms themselves are only a small part of the problem of providing recommendations to users.[2] We have a number of algorithms that work fairly well, and while there is room to refine them, there is much work to be done on user experience, data collection, and other problems which make up the whole of the recommender experience.[2]

1.3 History of digital Recommender:

Grundy [9], a computer based librarian, was an early step towards automatic recommender systems which was fairly primitive, grouping users into "stereotypes" based on a short interview and using hard-coded information about various stereotypes' book preferences to generate recommendations, but it represents an important early entry in the recommender systems space.[2]

In the early 1990s, collaborative filtering began to arise as a solution for dealing with overload in online information spaces.[2] Tapestry[10] was a manual collaborative filtering system that allowed the user to query for items in an information domain, such as corporate e-mail, based on other users' opinions or actions, for example the command "give me all the messages forwarded by John" returned the messages forwarded by John.[11][2]

Automated collaborative filtering systems soon followed, automatically locating relevant opinions and aggregating them to provide recommendations. GroupLens [12] used this technique to identify Usenet articles which are likely to be interesting to a particular user.[2] With these systems, users do not obtain any direct knowledge of other users' opinions, nor do they need to know what other users or items are in the system in order to receive recommendations.[2] This interest produced a number of recommender systems for various domains, such as Ringo[13] for music, the BellCore Video Recommender [14] for movies, and Jester [15] for jokes.

In the late 1990s, commercial deployments of recommender technology began to emerge. Perhaps the most widely-known application of recommender system technologies is Amazon.com.[16] Based on purchase history, browsing history, and the item a user is currently viewing, they recommend items for the user to consider purchasing.[16]

The toolbox of recommender techniques has also grown beyond collaborative filtering to include content-based approaches based on information retrieval, bayesian inference, and case-based reasoning methods [18] [19] [16]. Hybrid recommender systems [20] have also emerged as various recommender strategies have matured, combining multiple algorithms into composite systems that ideally build on the strengths of their component algorithms.[16]

Outside of computer science, the marketing literature has analyzed recommendations for its ability to increase sales and improve customer experience [16] [17].

1.4 Examples of recommender:

IMDB recommends movies, Goodreads recommends books, Amazon recommends products, Facebook's post on home page, etc.

1.5 Framework:

A programming framework is an abstraction that provides generic functionality, which can be used, modified or external functionality can be added by writing codes to fulfill user needs writing a minimal amount of code in minimum time. It prevents programmers from writing the same code over and over again by providing a generic template and popular functionality. Few leading frameworks are: Ruby on Rails, Django, etc.

1.6 Recommendation Framework:

Usually when a developer has to build a recommendation system he has to start writing code from scratch like on early days of web development, which consumes a lot of time and development energy. So in this thesis we are trying to separate the common tasks of every user user collaborative recommendation system as a template, and trying to provide a generic recommendation system based on the provided model at the same time to provide support for further development via API.

Chapter 2

Different types of Recommenders

2.1 Non-personalized Recommenders:

This type of recommendation system has to recommend without knowing anything about the user it is recommending to. It can be by comparing rating from anonymous or non anonymous users or people doing things together.
Different types of non personalized recommendations are:

2.1.1 User Rating:

Most popular example of non-personalized recommendation can be user rating. It takes the rating from the user and shows as a goodness measure to recommend a product.

There are many ways to represent user ratings and they are appropriate in different circumstances. Some of the examples are discussed below.

2.1.1.1 Average Rating

The simplest possible way of recommending the goodness of a product would be just showing the average of all user ratings.
For example Zagat, a famous restaurant review website acquired by google comes with four separate scores: food, decor, service, cost. Zagat calculates its scores by using avg, $round(mean(rating)*10)$

.

2.1.1.2 Rating count and distribution

In average rating one person's best rating gets diminished by another person's worst rating. So site like Crystal Cruises they takes user votes in term of binary form "good" or "awful". So user asks for whether certain cruise is recommended or not. So by looking at raiting we can tell percentage of people enjoying a certain cruse.

2.1.1.3 Damped mean:

Number of ratings tells us how confident we are about ratings. Initially when a product has low ratings we are not confident enough about its initial ratings. To solve low confidence problems we can use damped mean. In damped means without much evidence we assume everything is average.

$$\frac{\sum_u rating_u - k * \mu}{n+k}$$

2.1.1.4 Other approaches

Other than described methods, many websites use different rating mechanism

Hacker news uses this formula:

$$\frac{(U+D-1)^{\alpha}}{(t_{now}-t_{post})^{\gamma}} \times P \quad \alpha = 0.8, \gamma = 1.8$$

where P is hackernews' business logic scaling.

Raddit use completely differently

$$log_{10} max(1, |u-d|) + \frac{sign(u-d) \times t_{post}}{4500}$$

2.1.1.4 Cons of user rating

Rating has a problem that only users of two extreme levels tend to come back and vote products so it is self biased. When the quality of the product improves then the old votes are misleading. People may not like a product because of certain incidents like a restaurant without wifi.

2.1.2 Cross Selling: Recommendation depending on other product:

Some products are being bought together, for example people who bought cameras tend to buy a camera bag with it. So by recommending those products the sell can be increased. Usually cross selling is represented as “people who bought x also bought this”. Intuitively the formula should be:

$$\frac{x \cup y}{x}$$

where $x, y \in Products$, but it can happen that y is an extremely popular product, everybody buys it. In that case i won't be a good recommendation because they are going to buy that product anyway, this is popularly known as “Banana trap” because of the overly overwhelming popularity of bananas. So one way of filtering out this kind of product will be:

$$\frac{\frac{x \cup y}{x}}{\frac{!x \cup y}{!x}}$$

Or other popular data mining technique is assertion rule:

$$\frac{P(A \cup B)}{P(A)^{*} P(B)}$$

But the problem of association rules is that it is not direction specific. It basically calculates the co occurrence.

2.2 Content based recommenders:

Content based recommendation focus is mostly on content not on other users. Krakatoa chronicle could be an example of content based recommenders. It is an early news site made a combination of what users like and what the editor wants us to read. Because editors' choices are usually important events that you may like to be aware of.

2.2.1 Cons of content based recommendation

It requires model items in relevant attributes which requires human effort. Then collected attributes or keywords are made into vectors for mathematical operation which is often very costly but only one time operation. But it has many positive sides.

2.2.2 Pros of content based recommendation

Content based recommendation covers a wide variety of preferences. It can happen that the user himself is not aware of their preferences, for example in a movie he did not pay attention to its director. So automatic model generation will come handy. But the user should be able to edit his preferences. Once we have preference we can make a list of users likes and dislikes. Some time likes and dislikes won't suggest what user may like.

2.2.3 How to incorporate content based recommendation

The product that has description or while recommending news Tfidf is a common and a better way to find preferences. It filters out common keywords like "the" which appears very frequently in documents. Tfidf is defined as term frequency * inverse term frequency. Inverse term is usually defined in terms a log function looks like log (document/ document where term appeared). So the use case of tfidf in recommend is to build a profile for users liked items. That will be added to the user's preference vector. And for each document's property vector we can find out if the item should be recommended or not. Sometimes we may need to normalize tf as tf does not consider Document length. In okapi bm25 not only documents but it also considers frequency of terms in query. For phrases that contain multiple words may return unnecessary output because tfidf is doing on each word. For that bigram or trigram model may be useful. Terms appearing in the first paragraph are usually more important, we may also like to consider that.

2.3 Case based recommendation:

Case based recommendation is derived from case based reasoning of artificial intelligence where information is being retrieved from a structured database based on query but there are many ways of interaction. It works great when the user has a particular goal in mind.

2.3.1 Pros of case based recommendation:

Case based recommendation is simple. It requires less data. Case based recommendation is less likely to produce awkward results.

2.3.2 Cons of case based recommendation:

Problem on content based technique is that it requires attributes, attributes may not be present in some product. For example in painting the attributes are not present.

Chapter 3

Collaborative Recommendation

Collaborative filtering (CF) is one of the most popular recommendation algorithms. it provides predictions and recommendations based on the ratings of other users. Intuitively, they assume that, if users agree about the quality or relevance of some items, then they will likely agree about other items[2].

The majority of collaborative filtering algorithms in service today, operate by first generating predictions of the user's preference and then produce their recommendations by ranking candidate items by predicted preferences.[2]

Collaborative filtering can be of various types, such as user-user collaborative filtering, item-item collaborative filtering etc.

3.1 User-User Collaborative Recommendation:

User–user collaborative filtering, also known as k-NN collaborative filtering, was the first of the automated Collaborative Filtering methods.[2] User–user collaborative filtering was first introduced in the GroupLens Usenet article recommender[12]. BellCore video recommender[14], Ringo music recommender[13] uses user-user Collaborative Filtering (CF). Besides the rating matrix R, a user–user CF system requires a similarity function $s:U \times U \to R$ computing the similarity between two users and a method for using similarities and ratings to generate predictions. [2]

3.1.1 Pros of User-User Collaborative Recommendation:

User User Collaborative Filtering does not depend on error prone machine analysis of content .**[1]** It is able to filter any type of content, including images, texts, and music. It has been used in Groplens [31] [12], ringo [13], video recommender [14], movie lenz [1], amazon, movie finder, CD now, launch uses acf. **[1]** It is able to represent hard to represent concepts like taste and quality. **[1]** Acf is a stochastic process that computes models based on models that are heuristic approximations of human procesis. **[1]** Acf is done on very sparse and incomplete data. **[1]**

3.1.2 Cons of User-User Collaborative Recommendation:

User–user collaborative filtering suffers from scalability problems as the user base grows. [2] Searching for the neighbors of a user is an O(|U|) operation (or worse, depending on how similarities are computing — directly computing most similarity functions against all other users is linear in the total number of ratings).[2]

3.1.3 Predictors:

3.1.3.1 Non personalized personalized prediction:

In this prediction the output for user does not take part in the calculation of prediction.

$$P_{a,i} = \frac{\sum_{u=1}^{n} r_{u,i}}{n}$$

Where, P= prediction

U=user

R=rating

I=item

From this equation above we can move forward to two directions.

3.1.3.2 Personalized prediction:

The formula for personalized prediction is

$$P_{a,i} = \frac{\sum_{u=1}^{n} r_{u,i} * W_{u,I}}{\sum_{u=1}^{n} W_{u,I}}$$

The value of W can be varied from 0 to 1. We are setting W to be 1 for those user which is good for that user and we are filtering rest of the user by putting the value of W as 0. In the normalization term instead of n now we are using the sum of weighted function which prevents from incrementing the value of normalization terms when I am not actually taking the rating into consideration.

When the W function returns a negative value we need to fix this by using mod.

3.1.3.3 Mean centering the data:

Everyone has their own way of rating. Some users may rate 7 for the movies he likes the most, on the other hand some users may like 7 to the average movies he watches. So in this direction it is being considered.

$$P_{a},i = f_a + \frac{\sum_{u=1}^{n}(r_{u,i} - r_{avg,u})}{n}$$

3.1.3.4 Mean centering personalized prediction:

Putting both of these direction together:

$$P_{a,i} = f_a + \frac{\sum_{u=1}^{n}(r_{u,i} - r_a vg_u) * W_{u,I}}{\sum_{u=1}^{n}|W_{u,I}|}$$

3.1.3.5 Mean centering personalized normalization prediction:

Equation can also be extended to normalize user ratings to z-scores by dividing the offset from mean rating by the standard deviation σa of each user’s ratings, thereby compensating for users differing in rating spread as well as mean rating [22]:

$$P_{a,i} = f_a + \sigma_a \frac{\sum_{u=1}^{n}(z_{u,i} - r_a vg_u) * W_{u,I} / \sigma_u}{\sum_{u=1}^{n}|W_{u,I}|}$$ [2]

We can also normalize with z scores from means to detect whether users ratings are more sparse or not.

$$z_{a,i} = \frac{(r_{a,i} - r_{avg})}{\sigma_u}$$

Then by multiplying users sigma, we can try to convert it into users sparsity of choice. It can generate more rating value so sometimes we need to clamp it down.

3.1.3.6 Other prediction mechanisms

Not only weighted averaging there are other mechanisms that has been used for computing predictions. Ringo used no weighting, performing an unweighted average over ratings by neighboring users [13].[2] The BellCore system used a multivariate regression over the users in the neighborhood to generate predictions [14] .[2] Weighted averaging is by far the most common, however, as it is simple and works well in practice.[2] It is also consistent with Social Choice theory[32], grounding it in models of human behavior and an axiomatic, first principles development of collaborative filtering.[2] Social choice theory deals with the preferences of individuals and of a society as a whole.[2] Given several properties that a recommender should exhibit within the social choice framework, Pennock et al. show that weighted averaging is the only aggregation method which produces consistent results. [2]

3.1.4 Similarity functions:

3.1.4.1 Pearson correlation:

GroupLens and BellCore both used Pearson method [14] [12].[2] The correlation is computed by the following:[2]

$$W_{a,v} = \frac{\sum_{i=x}^{m} (r_{a,}i - r_{a,}avg)(r_{v,}i - r_{v,}avg)}{\sigma_a \sigma_v}$$

$$\sigma_u = \sqrt{\sum_{i \in I_u U I_v} (r_{u,i} - \overline{r_u})^2}$$

$$\sigma_v = \sqrt{\sum_{i \in I_u U I_v} (r_{v,i} - \overline{r_v})^2}$$

It basically the correlation where we get a higher weight for people who agrees a lot. When there is not much agreement it washed out the value to 0.

But if person has too less common items then the waiting function gets too high. So significance waiting is needed to be introduced.
So we will introduce a cut off for number of common items.

$$W_{a,u} = \frac{W_{a,u}}{min(n_{thr}, n_{common})}$$

Or bias values can be added with the base part.

But there are better ways to measure correlation in statistics that works better. Pearson method.

3.1.4.1.1 Cons of Pearson correlation:

Pearson correlation suffers from computing high similarity between users with few ratings in common. [2] This can be alleviated by setting a threshold on the number of co-rated items necessary for full agreement (correlation of 1) and scaling the similarity when the number of co-rated items falls below this threshold[22] [34]. [2] Experiments have shown a threshold value of 50 to be useful in improving prediction accuracy, and the threshold can be applied by multiplying the similarity function by min {|Iu∩Iv|/50,1}.[2]

3.1.4.2 Constrained Pearson correlation

Ringo solicited ratings from its users on a 7-point scale, providing a rating guide that fixed 4 as a neutral (neither like nor dislike) value rz.[2] With an absolute reference, it is possible to correlate absolute like/dislike rather than relative deviation (as the standard Pearson r does).[2] This led Shardanand and Maes[13] to propose the constrained Pearson correlation:[2]

$$W_{a,v} = \frac{\sum_{i=x}^{m} (r_{a},i - r_z)(r_{v},i - r_z)}{\sigma_a' \sigma_v'}$$

$$\sigma_u = \sqrt{\sum_{i \in I_u U I_v} (r_{u,i} - \overline{r_z})^2}$$

$$\sigma_v = \sqrt{\sum_{i \in I_u U I_v} (r_{v,i} - \overline{r_z})^2}$$

Another way could be by considering how much user likes an item than an avg user. In that case the prediction function would look like following:

$$\mu + \mu_i + \mu_u$$
$$\mu_i = \frac{\sum_u (r_{u,i} - \mu)}{n_i}$$
$$\mu_u = \frac{\sum_u (r_{u,I} - \mu - \mu_I)}{n_u}$$

3.1.4.3 Spearman rank correlation:

The Spearman rank correlation coefficient is another candidate for a similarity function[22].[2] For the Spearman correlation, the items a user has rated are ranked such that their highest-rated item is at rank 1 and lower rated items have higher ranks.[2] Items with the same rating are assigned the average rank for their position. The computation is then the same as that of the Pearson correlation, except that ranks are used in place of ratings.[2]

Another correlation function could be Spereman correlation function. Spereman correlation takes the ranking into consideration rather than the measurement.

3.1.4.4 Cosine similarity:

This model is somewhat different than the previously described approaches, as it is a vector-space approach based on linear algebra rather than a statistical approach.[2] Users are represented as |I|-dimensional vectors and similarity is measured by the cosine distance between two rating vectors.[2] This can be computed efficiently by taking their dot product and dividing it by the product of their L2 (Euclidean) norms: [2]

We can also find similarity using vector space model, cosine similarity:

$$W = \frac{r_u * r_v}{|r_u|^2 |r_v|^2} = \frac{\sum(r_{a,i} * r_{u,i})}{\sqrt{(\sum r_{a,i})^2}\sqrt{(\sum r_{u,i})^2}}$$

Now if we normalize rating then

$$W = \frac{\sum((r_{a,i} - \mu_a) * (r_{u,i} - \mu_u)}{\sqrt{(\sum r_{a,i} - \mu_a)^2}\sqrt{(\sum r_{u,i} - \mu_u)^2}}$$

We get the same formula for pearson correlation except it suggest a different way of handling items that are not common and also removes some of the need of significance waiting. So here we will assume that anything that has not been rated has a normalized rating of 0.

The threshold-based damping described for Pearson correlation can, in principle, be applied to any similarity function, although the benefits of doing so have not been studied.[2] An alternative method for damping is to add a large constant, such as 100, to the denominator; this attenuates the similarity of users with small sets of co-rated items. [2] Other similarity functions, such as mean-squared difference[13], have been proposed, but have not seen significant adoption.[2] In general, Pearson correlation has been found to provide the best results [35] [22], although results from the Ringo system suggest that constrained Pearson correlation may provide some improvement when items are rated on an absolute scale [13].[2]

3.1.4.5 Rule based similarity:

Does not usually use rule based filtering in acf. **[1]** MYCIN [36] used rule based reasoning of Lisp to English translation. **[1]**

3.2 Item–Item Collaborative Filtering

To extend collaborative filtering to large user bases and facilitate deployment on e-commerce sites, it was necessary to develop more scalable algorithms. [2] Item–item collaborative filtering, also called item-based collaborative filtering, takes a major step in this direction and is one of the most widely deployed collaborative filtering techniques today.[2] Item–item collaborative filtering was first described in the literature by Sarwar et al. [41] and Karypis [42], although a version of it seems to have been used by Amazon.com at this time [44].[2] Rather than using similarities between users' rating behavior to predict preferences, item– item CF uses similarities between the rating patterns of items.[2] If two items tend to have the same users like and dislike them, then they are similar and users are expected to have similar preferences for similar items. [2] In its overall structure, therefore, this method is similar to earlier content-based approaches to recommendation and personalization, but item similarity is deduced from user preference patterns rather than extracted from item data. [2] In its raw form, item–item CF does not fix anything: it is still necessary to find the most similar items (again solving the k-NN problem) to generate predictions and recommendations. [2] In a system that has more users than items, it allows the neighborhood-finding to be amongst the smaller of the two dimensions, but this is a small gain. [2] It provides major performance gains

by lending itself well to pre-computing the similarity matrix. [2] As a user rates and re-rates items, their rating vector will change along with their similarity to other users.[2] Finding similar users in advance is therefore complicated: a user's neighborhood is determined not only by their ratings but also by the ratings of other users, so their neighborhood can change as a result of new ratings supplied by any user in the system. [2] For this reason, most user–user CF systems find neighborhoods at the time when predictions or recommendations are needed. [2]
In systems with a sufficiently high user to item ratio, however, one user adding or changing ratings is unlikely to significantly change the similarity between two items, particularly when the items have many ratings. [2] Therefore, it is reasonable to pre-compute similarities between items in an item–item similarity matrix. [2] The rows of this matrix can even be truncated to only store the k most similar items. [2] As users change ratings, this data will become slightly stale, but the users will likely still receive good recommendations and the data can be fully updated by re-computing the similarities during a low-load time for the system.[2]

Item–item CF generates predictions by using the user's own ratings for other items combined with those items' similarities to the target item, rather than other users' ratings and user similarities as in user– user CF. [2] Similar to user–user CF, the recommender system needs a similarity function, this time s:I * I →R, a, and a method to generate predictions from ratings and similarities. [2]

Computing Predictions

In real-valued ratings domains, the similarity scores can be used to generate predictions using a weighted average, similar to the procedure used in user–user CF. [2] Recommendations are then generated by picking the candidate items with the highest predictions. [2] After collecting a set S of items similar to i, $p_{u,i}$ can be predicted as follows [44]: [2]

$$p_{u,i} = \frac{\sum_{j \in S} s(i,j) r_{u,j}}{\sum_{j \in S} |s(i,j)|}$$

S is typically selected to be the k items most similar to j that u has also rated for some neighborhood size k. Sarwar et al. [44] found k =30 produced good results on the MovieLens data set. Equation above suffers from two deficiencies. The first comes to light when it is possible for similarity scores to be negative and ratings are constrained to be nonnegative as some of the ratings averaged together to compute the prediction may be negative after weightings. While this will not affect the relative ordering of items by predicted value, it will bias the predicted values so they no longer map back to the user rating domain. This can be corrected either by thresholding similarities so only items with nonnegative similarities are considered or by averaging distance from the baseline predictors.

3.3 Explanation to improve collaborative recommendation:

In his keynote address at the 2009 ACM Conference on Recommender Systems, Martin [45] argued that the algorithms themselves are only a small part of the problem of providing recommendations to users.[2] We have a number of algorithms that work fairly well, and while there is room to refine them, there is much work to be done on user experience, data collection, and other problems which make up the whole of the recommender experience.[2] Acf proven to be working in entertainment.[31] [12] [14][13] For content domain where higher risk is associated, ACF did not prove its success. (eg. honeymoon tour). **[1]** So still the recommender could not gain user confidence.

Along with that it may have model **[1]** error where recommender user's specific context of requirement (eg. usually watch a movie but will go with his special friend) or data error where data given is not good enough. Data error can be of 3 kind **[1]**:

i) not enough data
ii) poor data and
iii) high variance data.

New or expensive items usually will have few ratings, so acf may have to be based on users who have less similarity. **[1]** There is one very common misconception among users that the ACF system is making decisions based on content characteristics. **[1]** Explanations can help either detect or estimate the likelihood of errors in the recommendation.

Explanation is helpful to users. When there is an explanation, it helps user in many way. Some of them are:

(1) Justification: user knows how much trust to do on that recommender **[1]**

(2) User Involvement: user also takes part in recommender data build up. **[1]**

(3) Education: user knows about strength and limitation of the recommender **[1]**

(4) Acceptance: Greater acceptance by users. **[1]**

Research investigation into measuring the filtering performance of users both with and without the explanation interface could not prove or disprove the hypothesis that explanations can increase the filtering performance due to many factors but 86% of these users said that they would like to see their explanation interface added to the system.**[1]** Adding explanation interfaces to an ACF system will improve the acceptance of that system among users.**[1]** It will make the filtering systems that are more accepted, more effective, more understandable, and which give greater control to the user.**[1]**

Explanation also solves many problems. Some time users mistake is considered as rating.**[1]** For example a user mistyped url or an unexpected pop up from sites ad which started recommender to recommend unexpected result. To resolve this we can give user freedom to choose what to recommend and to provide recommendation. MYCIN [36] is an example of recommender system that provides elaborate explanation.[1] MYCIN [36] used rule based reasoning of Lisp to English translation so user can ask how it draw a conclusion and what the recommender know.[1]
Other works [37] [31]

Johnson & Johnson [38] have begun research into the components of a unified theory
of explanation in human-computer interfaces.**[1]** They performed empirical experiments to help determine the logical components of an explanation. **[1]**
There has also been considerable study into the psychology of questioning and question answering with humans and how it can be applied to human-computer interfaces. [39] [40] [1]

But providing all information may not be a right solution, so there are several question that we might ask. **[1]**
What is the right amount of detail to expose?
How much information is too much?

What type of data do we present?

Conceptual model of explanation:

There can be two conceptual model of explanation:

i) White Box Conceptual Model

ii) Black Box Model

3.3.1 White Box Conceptual Model

Building model of explanation: The operation of an ACF system is analogous to the human word-of-mouth recommendation. **[1]**

3 Step process:

(1) User enters profile of ratings: We can collect an immense amount of preference information from the user, both implicit, such as page-views, and explicit, such as numeric ratings. The user can benefit from knowing how her actions have affected her recommendations. **[1]** There can have an interface and explanation of why it has been recommended.

(2) ACF system locates people with similar profiles (neighbors): ACF system has identify the correct set of neighbors for the user's current context of need. **[1] Question user may like to s**ee what does the profile look like of his neighbor or user may have a few special people that she wants as a neighbor or it can happen the opposite, user may not like certain user to be his neighbor. **[1]** Do their interests match the user's current context of information need. **[1]**

(3) Neighbors' ratings are combined to form recommendations. The final step is explaining the data and the process of taking the ratings of the neighbors and aggregating them into a final prediction. **[1]** At this level that many of the symptoms of weak predictions can be discovered with good explanations. **[1]**

Users can benefit greatly from knowing exactly how each of their neighbors rated the item being explained, or if there are large numbers of ratings, the distribution of ratings for the item being recommended. **[1]**

For example art film vs commercial film one will have less rating another will have huge rating, but art film having less number of votes and good rating means different things so it will be helpful for user if he knows about the tiny details.**[1]**

3.3.2 Black Box Model

Often, there is not the opportunity or possibly the desire to convey the conceptual model of the system to each user of the system.**[1]** In such cases, the ACF system becomes a black box recommender, producing recommendations like those of an oracle.**[1]** The user may not even be aware that the ACF system is collecting implicit ratings, such as time-spent-reading [31] to fuel the recommendations.**[1]**

One technique is to use the past performance of the recommender as justification.**[1]** For example, an explanation might include the statement "This system has been correct for you 80% of the time when recommending items similar to this one." **[1]** Another technique might be to bring in supporting evidence that may not have been used during the computation of the recommendation.**[1]** For example, even though the video store recommendation was based only on the purchase records of customers, the video store could justify its predictions by displaying correlating recommendations from local film critics.**[1]**

How to represent data to user:

Rating histograms seem to be the most compelling ways to explain the data behind a prediction. In addition, indications of past performance; comparisons to similar rated items; and domain specific content features, such the actors and actresses in a movie are also compelling ways to justify a high recommendation. [1]

Can explanation facilities increase the filtering performance of ACF system users? **[1]**

Chapter 3

Goodness measure of recommender

Over time researchers have developed many techniques to measure the goodness of recommenders.

4.1 Different techniques to measure recommender:

4.1.1. Accuracy and error measures:

Accuracy and error measure has been used in the early days. For calculating errors, MAE, RMSE, MSE were used. But all of these errors are related and produce identical results. Error from users who have 10 ratings and 1000 are considered equally. Error can dominate irrelevant parts of items.But for the recommendation system nobody cared about accuracy.

4.1.2 Decision-support metrics:

In evaluation of recommendation ROC AUC, Breese score has been used later precision and recall also used.

4.1.2.1 Precision:

Precision is the ratio of related item and selected item.

$$Percision = \frac{related\, item}{selected\, item}$$

For precision we need to test that what we recommend are all good or not. Precision figures out how well does the recommender do at finding good things.

4.1.2.2 Recall:

Recall is the measure of how many items are selected from related items. Motivation behind recall is that it tests all goods are recommended.

4.1.2.3 Percision and recall

We need both precision and recall in our recommendation system, so much we can mix both of them up in our evaluation. F matrice.

$$F = \frac{2pr}{(p+r)}$$

Precision and recall needs a ground truth, but if there is a ground truth then we will not be needing recommenders. So we may need to cover up some data to fake precision. Precision and recall takes the full dataset into consideration. Instead of that p@n means precession of first n items. Is widely used.

4.1.2.4 Mean average precision:

The Area under precision recall curve is the mean average precision.

4.1.2.5 ROC AUC:

ROC AUC plots of performance classifier at different thresholds. It plots true positive against false positive. In the recommendation system the curve reflects tradeoffs as we predicted cut off. Area under curve used as an effectiveness. ROC AUC can be used to tune recommender and to figure out "sweet spot"

4.1.3 Error meets decision-support/user experience:

It measures when a bad product appears in the top n recommended product. It does not usually used in research.

4.1.4 User-centered metrics:

It evaluates the coverage, user retention, recommendation uptake and overall user satisfaction.

4.2 Rank accuracy:

Rank accuracy is the measure about how well does the recommender estimate relative preference.

4.2.1 Mean Reciprocal Rank

Mean reciprocal rank figures out how many bad recommendations a user has to face to get its first good recommendation.

$$R = \frac{1}{i}$$

where i = the rank of the first 'good' item

4.2.2 Spearman Rank Correlation

Spearman rank correlation punishes all misplacement equally.

$$\rho = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}}$$

4.2.3 Discounted Cumulative Gain:

Things at front are more important than things are at back. **Discounted Cumulative Gain considers that. It** discounts by position, so things at front are more important than in the back. **It can be** normalized by total achievable utility, which is known as Normalized Discounted Cumulative Gain (nDCG)

4.2.4 Fraction of Concordant Pairs

Fraction of concordant pairs figures out the number of pairs that are in the correct relative order.

4.3 Things to be considered in evaluation:

Recommendation is not only about accuracy, there are various things that are to be considered while evaluating a recommender. In business nobody cares about accuracy, they want to increase the sales.

4.3.1 Diversity:

Users do not usually like to see all similar items in the recommendation just because he bought a similar item in the past. Common approach is to penalize/remove the items from the top-n list that are too similar to prior items already recommended just keeping the first item and replace with n+1st or later items – first ones that don't exhibit too high a similarity. So we may need to measure how different the items recommended are.

Diversification factor limits how many substitutions will be made

- **Clustering approaches** allow "diversification through bundling"
- **Scatter-gather** interfaces allow user controlled diversification
- **Business goal** don't turn away customers who are not currently interested in a narrow portion of your catalog

4.3.2 Serendipity:

It removes the obvious items from the list. Suggesting an item to the user in a surprise but that makes the user happy. For serendipity we need to simply downgrade items that are highly popular or otherwise obvious. It requires experimentation and tuning. It may use a diversification factor approach, or a constant down-weighting by popularity. Business goal also plays role in it, it gets people to consume less popular items

Chapter 5

Improving ACF with trusted category

User user collaborative filtering is a technique for recommendation which tries to establish a relationship between users and their rating based on other similar users rating patterns. The main hypothesis of user user collaborative filtering is that the user who has a similar previous rating will also like a similar rating. User user collaborative filtering works pretty well but yet it fails to explain certain cases. For example what if user A likes only scifi and action movies and hates romantic movies, another user B likes action and romantic movies. User A and B have 80% common in action movies. So the ACF algorithm will start suggesting romantic movies to user A. So in this paper we are suggesting several ways to solve this problem considering the category of movies.

User user collaborative filters basic prediction function looks like following:

$$P_{a,i} = \frac{\sum_{u=1}^{n} r_{u,i} * W_{u,I}}{\sum_{u=1}^{n} W_{u,I}}$$

5.1 Considering category in ACF

Instead of previous prediction function we suggest an enhanced prediction function which will take category into consideration.

$$p_{u,i} = \overline{r_u} + \sigma_u \frac{\sum_{u' \in U} S(u,u')(r_{u',i} - \overline{r_{u'}}) / \sigma'_u}{\sum_{u' \in U} S(u,u')}$$

$$p_{u,i} = \overline{r_u} + \sigma_u \frac{\sum_{u' \in U} S(u,u') S_c(u,i)(r_{u',i} - \overline{r_{u'}}) / \sigma'_u}{\sum_{u' \in U} S(u,u') S_c(u,i)}$$

$$p_{u,i} = \overline{r_u} + \frac{\sum_{u' \in U} S(u,u') S_c(u,i)(r_{u',i} - \overline{r_{u'}})}{\sum_{u' \in U} S(u,u') S_c(u,i)} \frac{\sigma_u}{\sigma'_u}$$

$S_c(u,i)$ is a function that computes whether this movie category is best suited for users or not. when the category is best suited it returns 1 and when it does not suit the user it returns 0. when it returns 1 the weight of user similarity is considered and when it returns 0 the weight of user similarity gets discarded.

The denominator is essentially for averaging the prediction. So when we discard the item we also need to subtract it from the denominator.

In this paper we have represented several ways to find category weight.

5.1.1 Category presence approach

We can filter out categories that are not rated by the user. in this approach we have a hypothesis that if a user interacts with a category then he is interested in that category of movies and we have clear ideas about whether he liked this category of movie or not.
For this approach we need to create users rated category as a 1 X n matrix. we can prepare this matrix by doing or operation in the category matrix of all movies user rated.

We will do an element wise multiplication between user category matrix U_c and I_c. and find out the max value of the matrix which basically returns 1 when there is a match between user category and item category.

$S_c(u,i) = max(I_c \cdot U_c)$

The limitation of this approach is that it fails to understand how much user like a certain category.

5.1.2 category interest approach

Category presence approach solves the problem of recommending unwanted category items to users but it assumes the user likes all categories equally. Practically it is very usual that users like different categories on a different scale. In this approach we tried to figure out users' category interest from her rating.

User and category weight can be computed by summing up the how many times user rated that category divided by maximum rating of a category.

$$S_c(u,c) = \frac{\sum_{m \in M} C_{c,m}}{max \bigcup_{c_1 \in C} (\sum_{m \in M} C_{c_1,m})}$$

For example if we have $m_0 \cdots m_2$ movies of category $c_0 \cdots c_2$then:

	c_0	c_1	c_2
m_0	1	0	0
m_1	1	1	0
m_2	0	1	1
	2	2	1

Table 4.1.2: example of category interest approach

$S_c(u,c2)$=1/2=0.5

5.1.3 category interaction distance approach

category presence or category interest based approach filters out all the movies of category the user did not rated does not matter how good those movies are or how well the similar user has rated it. In practical life users may like to experiment with the movie category as it is not in the risk domain. In this approach we will try to figure out the distance of items category with respect to users rated category based on others category choice. To figure out that we need to iterate through all the categories the movie belongs and all the categories user rated and figure our the distance between them in terms of category. When there is a common category it should return 1 but when there is no match it should return a probable function user liking it based on other user:

$$S_c(U_c, M_c) = max \bigcup_{u_c \in Uc} \bigcup_{m_c \in M_c} C_{u_c, m_c}$$

Where,

$$C_{i,j} = \frac{|user\ with\ i\ and\ j\ ratinng|}{|user\ with\ i\ or\ j\ rating|}$$

5.1.4 category Likelihood distance approach

Category presence approach, category interest approach and category interaction distance approach assumes that users' bad rating will fade away category similarity when the user won't like an item. it assumes that does not matter whether likes or hates a movie, as long as he rated the movie she liked the category. In this approach we would consider how much a user really liked not only rated but also liked.
probability of the user liking category given for an item is calculated.

L_c = number of movies of c category above threshold rating
D_c= number of movies of c category above threshold rating

$$C_{i,j} = \frac{\frac{L_i}{L_i + D_i} * \frac{L_j}{L_j + D_j}}{\frac{L_i + L_j}{L_i + D_i + L_j + D_j}}$$

Chapter 6

Result and Discussion

6.1 Result from category consideration approaches

Considering trusted category approach on MovieLens data set (http://grouplens.org/datasets/movielens/) over 100,000 ratings from 1000 users on 1700 movies we get the following result:

	Pearson correlation	Spearman rank correlation
Without category consideration	13.1540210919	0.0296703526013
category presence approach	13.1546702841	0.0296699456852
category interest approach	13.1546697499	0.0296699700349
category interaction distance approach	13.15469303	0.0296700345427

Table 5.1: Result from trusted category approach

6.2 Discussion on result

From the result above we can see that in terms of Pearson correlation category consideration approaches are not doing well but in terms of Spearman rank correlation all of the category consideration approaches are doing well. So we can come to the conclusion that in ranking if we consider the user's taste of the category then we can get better ranking in recommendation.

References

[1] Herlocker, JL. "Explaining Collaborative Filtering Recommendations." 2000. <http://grouplens.org/site-content/uploads/explain-CSCW-20001.pdf>

[2] Ekstrand, Michael D, John T Riedl, and Joseph A Konstan. "Collaborative filtering recommender systems." *Foundations and Trends in Human-Computer Interaction* 4.2 (2011): 81-173

[3]Shuo Chang, F. Maxwell Harper, and Loren Terveen. 2015. Using Groups of Items for Preference Elicitation in Recommender Systems. *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work and Social Computing*, ACM, 1258–1269. http://doi.org/10.1145/2675133.2675210

[4] Amazon.com, "Q4 2009 Financial Results," Earnings Report Q4-2009, January 2010

[5] Bennett, James, and Stan Lanning. "The netflix prize." *Proceedings of KDD cup and workshop* 12 Aug. 2007: 35.

[6] Adomavicius, Gediminas, and Alexander Tuzhilin. "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions." *Knowledge and Data Engineering, IEEE Transactions on* 17.6 (2005): 734-749.

[7] Su, Xiaoyuan, and Taghi M Khoshgoftaar. "A survey of collaborative filtering techniques." *Advances in artificial intelligence* 2009 (2009): 4

[8] F. J. Martin, "RecSys '09 industrial keynote: Top 10 lessons learned developing deploying and operating real-world recommender systems," in ACM RecSys '09, pp. 1–2, ACM, 2009.

[9] Rich, Elaine. "User modeling via stereotypes*." *Cognitive science* 3.4 (1979): 329-354.

[10] Goldberg, David et al. "Using collaborative filtering to weave an information tapestry." *Communications of the ACM* 35.12 (1992): 61-70

[11] Schafer, J Ben et al. "Collaborative filtering recommender systems." *The adaptive web* (2007): 291-324.

[12] Resnick, Paul et al. "GroupLens: an open architecture for collaborative filtering of netnews." *Proceedings of the 1994 ACM conference on Computer supported cooperative work* 22 Oct. 1994: 175-186.
[13] Shardanand, Upendra, and Pattie Maes. "Social information filtering: algorithms for automating “word of mouth”." *Proceedings of the SIGCHI conference on Human factors in computing systems* 1 May. 1995: 210-217.
[14] Hill, Will et al. "Recommending and evaluating choices in a virtual community of use." *Proceedings of the SIGCHI conference on Human factors in computing systems* 1 May. 1995: 194-201.
[15] Goldberg, Ken et al. "Eigentaste: A constant time collaborative filtering algorithm." *Information Retrieval* 4.2 (2001): 133-151.
[16] Ansari, Asim, Skander Essegaier, and Rajeev Kohli. "Internet recommendation systems." *Journal of Marketing research* 37.3 (2000): 363-375.
[17] West, Patricia M et al. "Agents to the Rescue?." *Marketing letters* 10.3 (1999): 285-300.
[18] Schafer, J Ben, Joseph A Konstan, and John Riedl. "E-commerce recommendation applications." *Applications of Data Mining to Electronic Commerce* (2001): 115-153.
[19] Smyth, Barry. "Case-based recommendation." *The adaptive web* (2007): 342-376.
[20] Burke, Robin. "Hybrid recommender systems: Survey and experiments." *User modeling and user-adapted interaction* 12.4 (2002): 331-370.
[21] Schafer, J Ben et al. "Collaborative filtering recommender systems." *The adaptive web* (2007): 291-324.
[22] Herlocker, Jon, Joseph A Konstan, and John Riedl. "An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms." *Information retrieval* 5.4 (2002): 287-310.
[23] Koren, Yehuda. "Factorization meets the neighborhood: a multifaceted collaborative filtering model." *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* 24 Aug. 2008: 426-434.
[24] Paterek, Arkadiusz. "Improving regularized singular value decomposition for collaborative filtering." *Proceedings of KDD cup and workshop* 12 Aug. 2007: 5-8.
[25] Potter, Gavin. "Putting the collaborator back into collaborative filtering." *Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition* 24 Aug. 2008: 3.

[26] Bell, Robert M, and Yehuda Koren. "Scalable collaborative filtering with jointly derived neighborhood interpolation weights." *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on* 28 Oct. 2007: 43-52.

[27] Herlocker, Jon, Joseph A Konstan, and John Riedl. "An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms." *Information retrieval* 5.4 (2002): 287-310.

[28] Potter, Gavin. "Putting the collaborator back into collaborative filtering." *Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition* 24 Aug. 2008: 3.

[29] Funk, Simon. "Netflix update: Try this at home." Dec. 2006.

[30] Koren, Yehuda. "Collaborative filtering with temporal dynamics." *Communications of the ACM* 53.4 (2010): 89-97.

[31] Konstan, Joseph A et al. "GroupLens: applying collaborative filtering to Usenet news." *Communications of the ACM* 40.3 (1997): 77-87.

[32] Pennock, David M, Eric Horvitz, and C Lee Giles. "Social choice theory and recommender systems: Analysis of the axiomatic foundations of collaborative filtering." *AAAI/IAAI* 30 Jul. 2000: 729-734.

[33] Lathia, Neal, Stephen Hailes, and Licia Capra. "Temporal collaborative filtering with adaptive neighbourhoods." *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval* 19 Jul. 2009: 796-797.

[34] Herlocker, Jonathan L et al. "An algorithmic framework for performing collaborative filtering." *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval* 1 Aug. 1999: 230-237.

[35] Breese, John S, David Heckerman, and Carl Kadie. "Empirical analysis of predictive algorithms for collaborative filtering." *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence* 24 Jul. 1998: 43-52.

[36] Buchanan, BG. "Rule Based Expert Systems: The Mycin Experiments of the ..." 1984. <http://dl.acm.org/citation.cfm?id=1096485>

[37] Horvitz, Eric J, John S Breese, and Max Henrion. "Decision theory in expert systems and artificial intelligence." *International journal of approximate reasoning* 2.3 (1988): 247-302.

[38] Johnson, Hilary, and Peter Johnson. "Explanation facilities and interactive systems." *Proceedings of the 1st international conference on Intelligent user interfaces* 1 Feb. 1993: 159-166.

[39] Reviewer-White, JS. "Review of Questions and information systems by Thomas W ..." 1993. <http://dl.acm.org/citation.cfm?id=972483>

[40] "Amazon.com: The Psychology of Questions ..." 2012. 19 Sep. 2015 <http://www.amazon.com/The-Psychology-Questions-Arthur-Graesser/dp/0898594448>

[41] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Reidl, “Item-based collaborative filtering recommendation algorithms,” in ACM WWW ’01, pp. 285–295, ACM, 2001.

[42] G. Karypis, “Evaluation of item-based top-N recommendation algorithms,” in ACM CIKM ’01, pp. 247–254, ACM, 2001.

[43] G. Linden, B. Smith, and J. York, “Amazon.com recommendations: Item-to-item collaborative filtering,” IEEE Internet Computing, vol. 7, no. 1, pp. 76–80, 2003.

[44] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Reidl, “Item-based collaborative filtering recommendation algorithms,” in ACM WWW ’01, pp. 285–295,ACM, 2001.

[45] F. J. Martin, “RecSys ’09 industrial keynote: Top 10 lessons learned developing deploying and operating real-world recommender systems,” in ACM RecSys’09, pp. 1–2, ACM, 2009.

[46] https://en.wikipedia.org/wiki/Microservices[1] Herlocker, JL. "Explaining Collaborative Filtering Recommendations." 2000. <http://grouplens.org/site-content/uploads/explain-CSCW-20001.pdf>

[2] Ekstrand, Michael D, John T Riedl, and Joseph A Konstan. "Collaborative filtering recommender systems." *Foundations and Trends in Human-Computer Interaction* 4.2 (2011): 81-173

[3]Shuo Chang, F. Maxwell Harper, and Loren Terveen. 2015. Using Groups of Items for Preference Elicitation in Recommender Systems. *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work and Social Computing*, ACM, 1258–1269. http://doi.org/10.1145/2675133.2675210

[4] Amazon.com, “Q4 2009 Financial Results,” Earnings Report Q4-2009, January 2010

[5] Bennett, James, and Stan Lanning. "The netflix prize." *Proceedings of KDD cup and workshop* 12 Aug. 2007: 35.
[6] Adomavicius, Gediminas, and Alexander Tuzhilin. "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions." *Knowledge and Data Engineering, IEEE Transactions on* 17.6 (2005): 734-749.
[7] Su, Xiaoyuan, and Taghi M Khoshgoftaar. "A survey of collaborative filtering techniques." *Advances in artificial intelligence* 2009 (2009): 4

[8] F. J. Martin, “RecSys ’09 industrial keynote: Top 10 lessons learned developing deploying and operating real-world recommender systems,” in ACM RecSys ’09, pp. 1–2, ACM, 2009.
[9] Rich, Elaine. "User modeling via stereotypes*." *Cognitive science* 3.4 (1979): 329-354.
[10] Goldberg, David et al. "Using collaborative filtering to weave an information tapestry." *Communications of the ACM* 35.12 (1992): 61-70
[11] Schafer, J Ben et al. "Collaborative filtering recommender systems." *The adaptive web* (2007): 291-324.
[12] Resnick, Paul et al. "GroupLens: an open architecture for collaborative filtering of netnews." *Proceedings of the 1994 ACM conference on Computer supported cooperative work* 22 Oct. 1994: 175-186.
[13] Shardanand, Upendra, and Pattie Maes. "Social information filtering: algorithms for automating “word of mouth”." *Proceedings of the SIGCHI conference on Human factors in computing systems* 1 May. 1995: 210-217.
[14] Hill, Will et al. "Recommending and evaluating choices in a virtual community of use." *Proceedings of the SIGCHI conference on Human factors in computing systems* 1 May. 1995: 194-201.
[15] Goldberg, Ken et al. "Eigentaste: A constant time collaborative filtering algorithm." *Information Retrieval* 4.2 (2001): 133-151.
[16] Ansari, Asim, Skander Essegaier, and Rajeev Kohli. "Internet recommendation systems." *Journal of Marketing research* 37.3 (2000): 363-375.
[17] West, Patricia M et al. "Agents to the Rescue?." *Marketing letters* 10.3 (1999): 285-300.
[18] Schafer, J Ben, Joseph A Konstan, and John Riedl. "E-commerce recommendation applications." *Applications of Data Mining to Electronic Commerce* (2001): 115-153.
[19] Smyth, Barry. "Case-based recommendation." *The adaptive web* (2007): 342-376.

[20] Burke, Robin. "Hybrid recommender systems: Survey and experiments." *User modeling and user-adapted interaction* 12.4 (2002): 331-370.
[21] Schafer, J Ben et al. "Collaborative filtering recommender systems." *The adaptive web* (2007): 291-324.
[22] Herlocker, Jon, Joseph A Konstan, and John Riedl. "An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms." *Information retrieval* 5.4 (2002): 287-310.
[23] Koren, Yehuda. "Factorization meets the neighborhood: a multifaceted collaborative filtering model." *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* 24 Aug. 2008: 426-434.
[24] Paterek, Arkadiusz. "Improving regularized singular value decomposition for collaborative filtering." *Proceedings of KDD cup and workshop* 12 Aug. 2007: 5-8.
[25] Potter, Gavin. "Putting the collaborator back into collaborative filtering." *Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition* 24 Aug. 2008: 3.
[26] Bell, Robert M, and Yehuda Koren. "Scalable collaborative filtering with jointly derived neighborhood interpolation weights." *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on* 28 Oct. 2007: 43-52.
[27] Herlocker, Jon, Joseph A Konstan, and John Riedl. "An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms." *Information retrieval* 5.4 (2002): 287-310.
[28] Potter, Gavin. "Putting the collaborator back into collaborative filtering." *Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition* 24 Aug. 2008: 3.
[29] Funk, Simon. "Netflix update: Try this at home." Dec. 2006.
[30] Koren, Yehuda. "Collaborative filtering with temporal dynamics." *Communications of the ACM* 53.4 (2010): 89-97.
[31] Konstan, Joseph A et al. "GroupLens: applying collaborative filtering to Usenet news." *Communications of the ACM* 40.3 (1997): 77-87.
[32] Pennock, David M, Eric Horvitz, and C Lee Giles. "Social choice theory and recommender systems: Analysis of the axiomatic foundations of collaborative filtering." *AAAI/IAAI* 30 Jul. 2000: 729-734.

[33] Lathia, Neal, Stephen Hailes, and Licia Capra. "Temporal collaborative filtering with adaptive neighbourhoods." *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval* 19 Jul. 2009: 796-797.

[34] Herlocker, Jonathan L et al. "An algorithmic framework for performing collaborative filtering." *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval* 1 Aug. 1999: 230-237.

[35] Breese, John S, David Heckerman, and Carl Kadie. "Empirical analysis of predictive algorithms for collaborative filtering." *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence* 24 Jul. 1998: 43-52.

[36] Buchanan, BG. "Rule Based Expert Systems: The Mycin Experiments of the ..." 1984. <http://dl.acm.org/citation.cfm?id=1096485>

[37] Horvitz, Eric J, John S Breese, and Max Henrion. "Decision theory in expert systems and artificial intelligence." *International journal of approximate reasoning* 2.3 (1988): 247-302.

[38] Johnson, Hilary, and Peter Johnson. "Explanation facilities and interactive systems." *Proceedings of the 1st international conference on Intelligent user interfaces* 1 Feb. 1993: 159-166.

[39] Reviewer-White, JS. "Review of Questions and information systems by Thomas W ..." 1993. <http://dl.acm.org/citation.cfm?id=972483>

[40] "Amazon.com: The Psychology of Questions ..." 2012. 19 Sep. 2015 <http://www.amazon.com/The-Psychology-Questions-Arthur-Graesser/dp/0898594448>

[41] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Reidl, “Item-based collabora-tive filtering recommendation algorithms,” in ACM WWW ’01, pp. 285–295, ACM, 2001.

[42] G. Karypis, “Evaluation of item-based top-N recommendation algorithms,” in ACM CIKM ’01, pp. 247–254, ACM, 2001.

[43] G. Linden, B. Smith, and J. York, “Amazon.com recommendations: Item-to-item collaborative filtering,” IEEE Internet Computing, vol. 7, no. 1, pp. 76–80, 2003.

[44] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Reidl, “Item-based collabora-tive filtering recommendation algorithms,” in ACM WWW ’01, pp. 285–295,ACM, 2001.

[45] F. J. Martin, “RecSys ’09 industrial keynote: Top 10 lessons learned developing deploying and operating real-world recommender systems,” in ACM RecSys’09, pp. 1–2, ACM, 2009.

[46] https://en.wikipedia.org/wiki/Microservices